Space Complexity of Fast D-Finite Function Evaluation

Marc Mezzarobba

Pequan

RAIM 2013

Binary Splitting

A classical method to evaluate series of rational numbers

- Classical constants
  Ex.: $$\frac{1}{\pi} = 12 \sum_{k=0}^{\infty} \frac{(-1)^k\,(6k)!\,(13591409 + 545140134k)}{(3k)!\,(k!)^3\,640320^{3k+3/2}}$$
  [Chudnovsky & Chudnovsky 1989]
- Elementary functions [Brent 1976, ...]
  $$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$
- D-Finite functions

D-Finite Functions

[Stanley 1980, Zeilberger 1990, ...]

An analytic function $y(z) : \mathbb{C} \to \mathbb{C}$ is D-finite (holonomic) iff it satisfies a linear (homogeneous) ODE with polynomial coefficients:

$$a_r(z)\, y^{(r)}(z) + \cdots + a_1(z)\, y'(z) + a_0(z)\, y(z) = 0, \quad a_j \in \mathbb{K}[z].$$

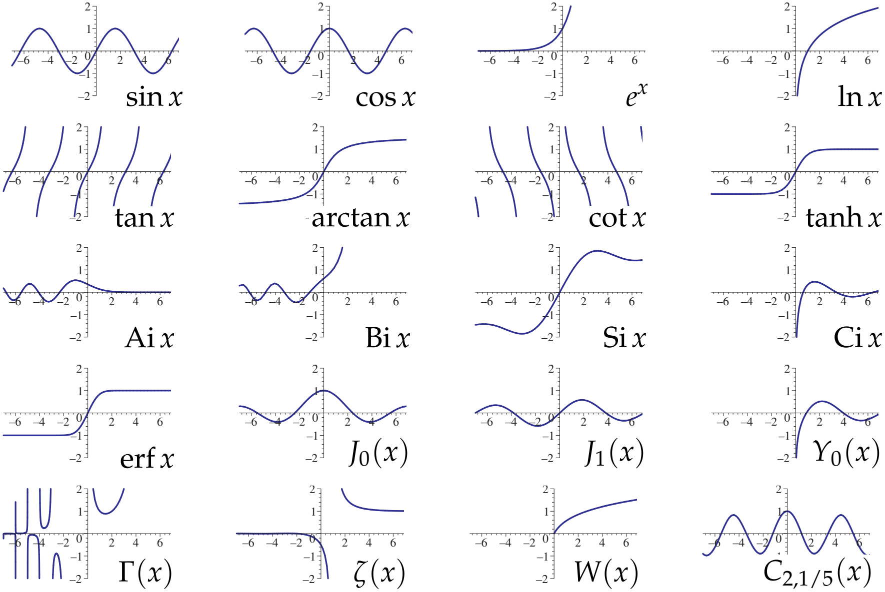
The sequence of Taylor coefficients of a D-finite functions obeys a linear recurrence relation with polynomial coefficients.

Example: $y(z) = \sum_{n \geq 0} y_n z^n = \exp z$

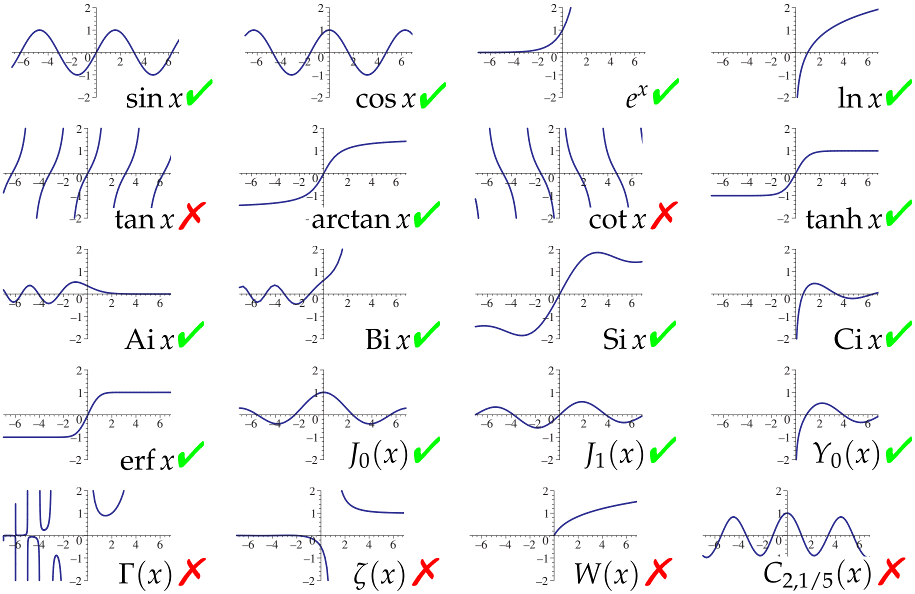
$$y'(z) - y(z) = 0 \qquad y(0) = 1$$

$$(n+1)\, y_{n+1} - y_n = 0 \qquad y_0 = 1$$

Elementary and Special Functions



Elementary and Special Functions



Main Result

Theorem

"D-finite functions can be evaluated in quasi-linear time and linear space."

That is: Fix a D-finite function y and a point $\zeta \in \mathbb{C}$.
The value $y(\zeta)$ may be computed with absolute error $\leq 2^{-d}$ in $O(M(d)(\log d)^2)$ operations, using $O(d)$ bits of memory.

Here $M(n)$ = compl. of $\leq n$-bit integer mult. $= O(n(\log n)e^{O(\log^* n)})$

Previous results

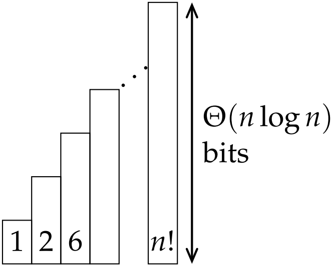
Same time complexity, space $O(d \log d)$
[Chudnovsky & Chudnovsky 1990, van der Hoeven 1999, M. 2010]

Space $O(d)$ in special cases [Brent 1976, …, Yakhontov 2011]

Warm-Up: Computing $n!$

Naïve algorithm

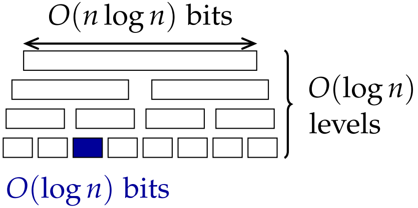
$$n! = n \times (n-1)!$$



Time: $\Omega(n^2 \log n)$
Space: $O(n \log n)$

Binary splitting (= product tree)

$$n! = (\lfloor n/2 \rfloor \cdots n) \times (1 \cdots \lfloor n/2 \rfloor)$$



Time: $O(M(n \log n) \log n)$
Space: $O(n \log n)$

Standard assumption: $M(n+m) \leq M(n) + M(m)$.

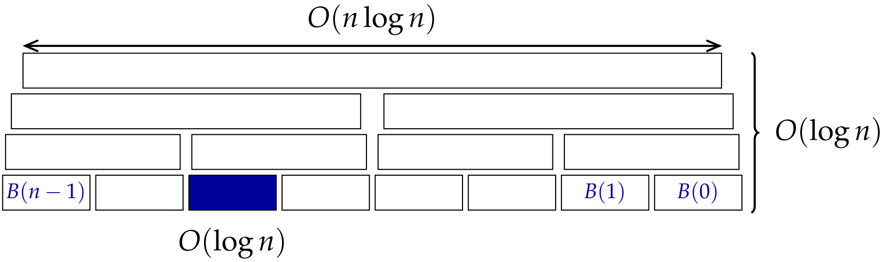
In the special case of $n!$: time $O(M(n \log n))$ [Borwein, Schönhage].

Binary Splitting for D-Finite Functions

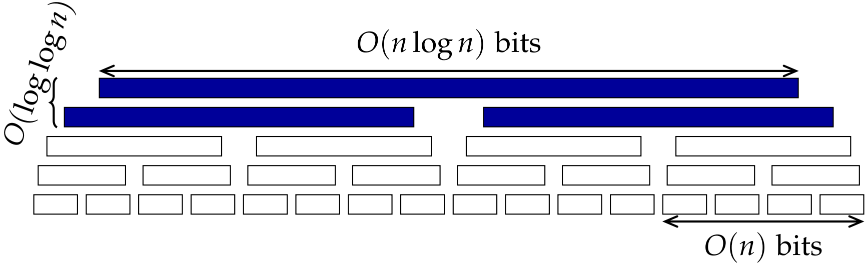
Same idea, using a matrix product tree

$$y(z) = \sum_{n=0}^{\infty} y_n z^n \qquad Y_{n+1} = C(n) Y_n, \quad Y_n = (y_n, \ldots, y_{n+s-1})^{\mathrm{T}}$$

$$S_n = \sum_{k=0}^{n-1} y_k \zeta^k \qquad \begin{bmatrix} Y_{n+1} \\ S_{n+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \zeta C(n) & 0 \\ 1, 0, \ldots & 1 \end{bmatrix}}_{B(n)} \begin{bmatrix} Y_n \\ S_n \end{bmatrix}$$

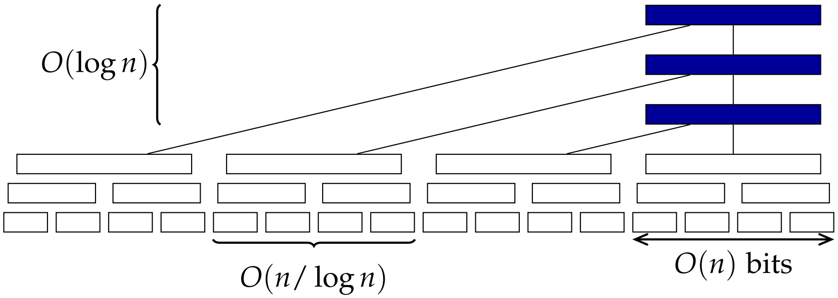


The Truncation Trick [Gourdon & Sebah, 1996?]



The Truncation Trick

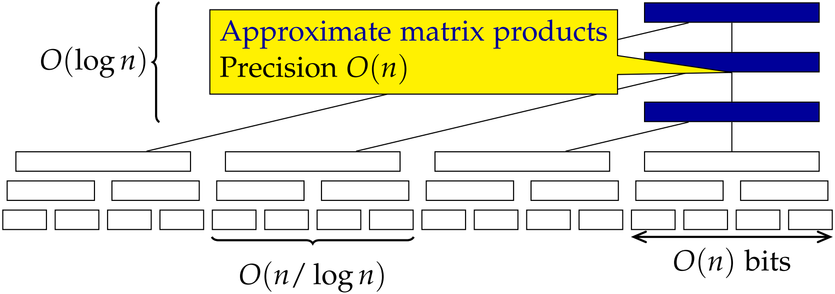
[Gourdon & Sebah, 1996?]



Time: $O(M(n \log n) \log n)$ (as before)
Space: $O(n)$

The Truncation Trick

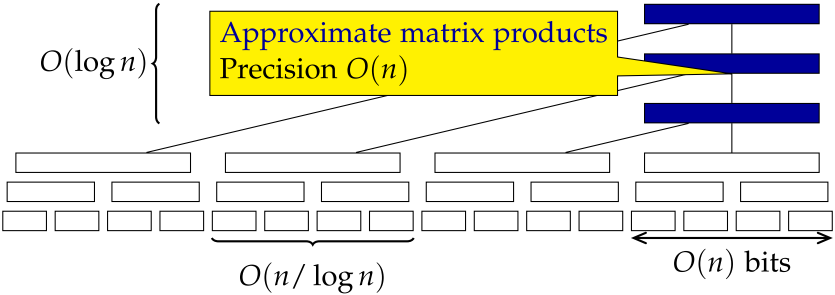
[Gourdon & Sebah, 1996?]



Time: $O(M(n \log n) \log n)$ (as before)
Space: $O(n)$

The Truncation Trick

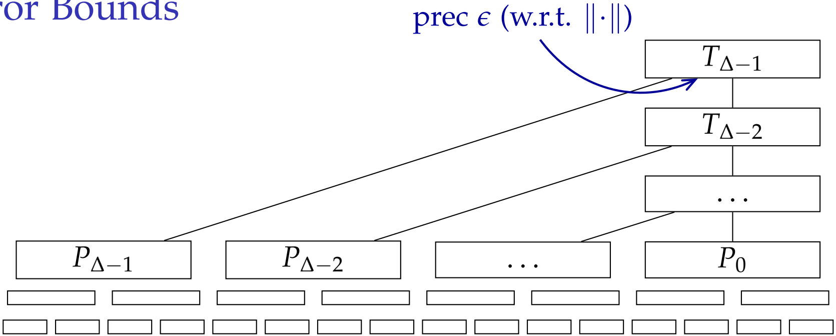
[Gourdon & Sebah, 1996?]



Time: $O(M(n \log n) \log n)$ (as before)
Space: $O(n)$

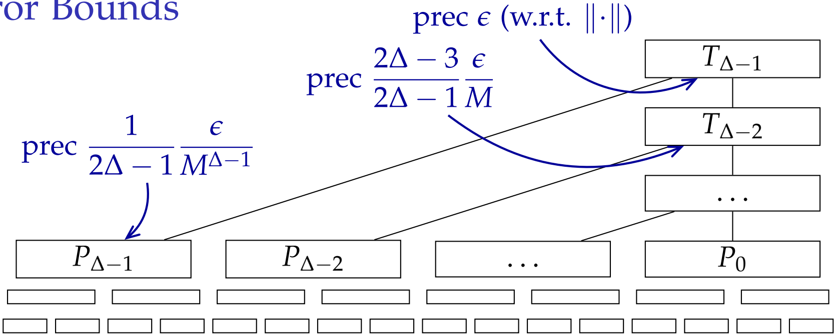
...provided the roundoff errors do not interfere!
[Yakhontov 2011 — Hypergeometric case]

Error Bounds



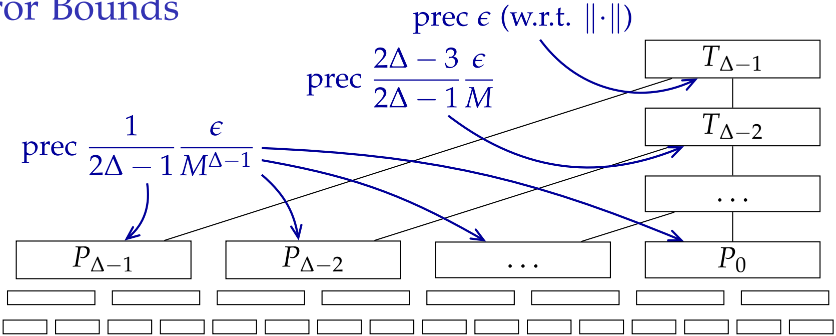
- Assume $\|P_0\|, \|P_1\|, \ldots \leq M$

Error Bounds



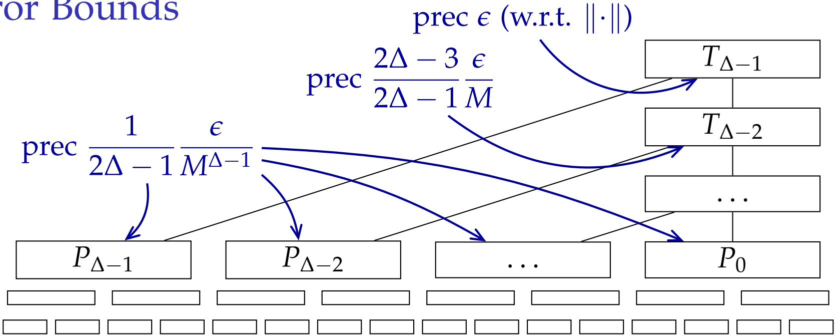
- Assume $\|P_0\|, \|P_1\|, \ldots \leq M$
- $\left\|\tilde{Q}\tilde{P} - QP\right\| \leq \left\|\tilde{Q} - Q\right\| \|P\| + \left\|\tilde{Q}\right\| \left\|\tilde{P} - P\right\|$

Error Bounds



- Assume $\|P_0\|, \|P_1\|, \ldots \leq M$
- $\|\tilde{Q}\tilde{P} - QP\| \leq \|\tilde{Q} - Q\| \|P\| + \|\tilde{Q}\| \|\tilde{P} - P\|$

Error Bounds



- Assume $\|P_0\|, \|P_1\|, \ldots \leq M$
- $\|\tilde{Q}\tilde{P} - QP\| \leq \|\tilde{Q} - Q\| \, \|P\| + \|\tilde{Q}\| \, \|\tilde{P} - P\|$
- $P_i = B(\lfloor \frac{i+1}{\Delta} n \rfloor - 1) \cdots B(\lfloor \frac{i}{\Delta} n \rfloor)$, so $M \leq C^{\lceil n/\Delta \rceil}$
- Max working prec $\approx \frac{1}{2\Delta} \frac{\epsilon}{M^\Delta} \gtrsim \frac{1}{2\Delta} \frac{\epsilon}{C^n}$,
  i.e., $\lesssim \log \epsilon^{-1} + n \log C + o(n) = O(n)$ digits

Prototype Implementation in NumGfun

```
> diffeq := collect({diff(y(z),z,z)-(-2*
z^5+4*z^3+z^4*a-2*z-a)*(diff(y(z),z))/(
(z-1)^3*(z+1)^3)-(-z^2*b+(-c-2*a)*z-d)*y
(z)/((z-1)^3*(z+1)^3),y(0)=1,D(y)(0)=0},
diff,factor);
```

$$diffeq := \left\{ \frac{d^2}{dz^2}\,y(z) - \frac{(-2z^3+z^2a+2z+a)\left(\frac{d}{dz}\,y(z)\right)}{(z+1)^2(z-1)^2} + \frac{(z^2b+zc+2za+d)\,y(z)}{(z-1)^3(z+1)^3},\ y(0)=1,\ D(y)(0)=0 \right\}$$

```
> a, b, c, d := 1, 1/3, 1/2, 3;
```

$$a, b, c, d := 1, \frac{1}{3}, \frac{1}{2}, 3$$

```
> evalf[51](HeunD(a, b, c, d, 1/3));
```

1.23715744756395253918007831405821000395447403052069

```
> myHeunD := diffeqtoproc(diffeq, y(z)):
> myHeunD(1/3, 50);
```

1.23715744756395253918007831405821000395447403052075

```
> diffeq := random_diffeq(3, 2);
```

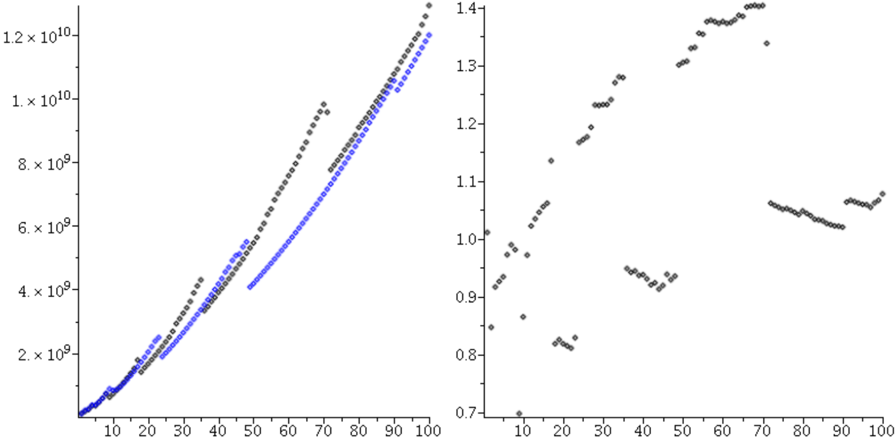
$$diffeq := \left\{ \left(\frac{13}{30}+\frac{8}{15}z+\frac{7}{30}z^2\right)y(z) + \left(-\frac{9}{20}+\frac{29}{30}z - \frac{1}{12}z^2\right)\left(\frac{d}{dz}\,y(z)\right) + \left(-\frac{43}{60}+\frac{49}{60}z + \frac{11}{30}z^2\right)\left(\frac{d^2}{dz^2}\,y(z)\right) + \left(-\frac{7}{12}+\frac{17}{30}z - \frac{3}{5}z^2\right)\left(\frac{d^3}{dz^3}\,y(z)\right),\ y(0)=0,\ D(y)(0)=\frac{7}{30},\ D^{(2)}(y)(0) = -\frac{43}{60} \right\}$$

```
> evaldiffeq(diffeq, y(z), (1+I)/5, 40);
```

0.04485557487767843131893308147593115486663 + 0.01990489830212805305047897725810997882821 I

`http://algo.inria.fr/libraries/papers/gfun.html`
(GNU LGPL)

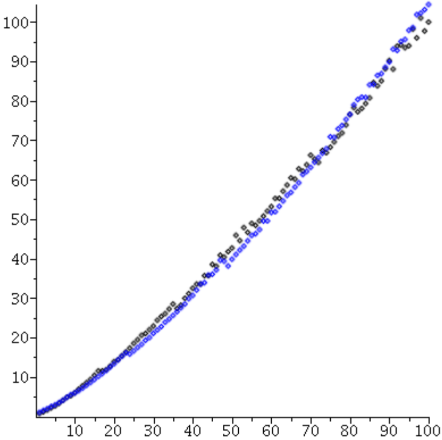
(Very Preliminary) Experimental Results: Space

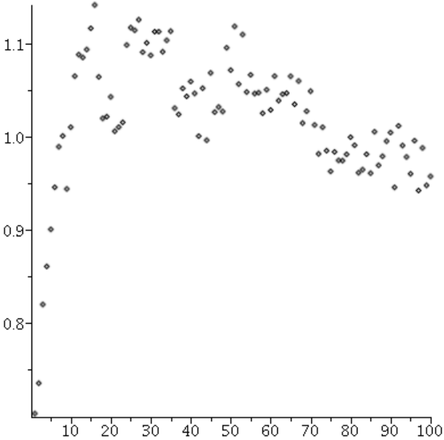


Diff. eq. from previous slide, $z = 1/5$, prec $= 1\,000, 2\,000, \ldots, 100\,000$

Left: black = classical, blue = truncated Right : classical/truncated

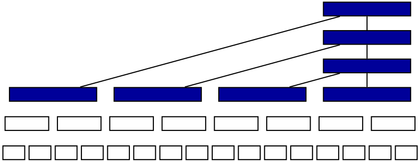
(Very Preliminary) Experimental Results: Time





Summary

- D-Finite functions may be evaluated in quasi-linear time and linear space
- Proof based on Chudnovsky & Chudnovsky's binary splitting algorithm
  + effective error bounds
- Prototype implementation available (pure Maple)

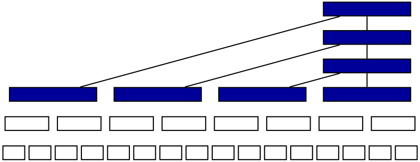


Current & future work

- Make it practical!
- Seems to require a more careful / lower-level implementation

Summary

- D-Finite functions may be evaluated in quasi-linear time and linear space
- Proof based on Chudnovsky & Chudnovsky's binary splitting algorithm
  + effective error bounds
- Prototype implementation available (pure Maple)

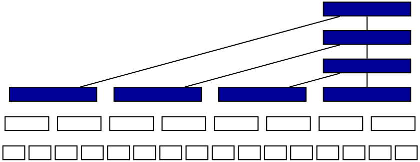


Current & future work

- Make it practical!
- Seems to require a more careful / lower-level implementation

Summary

- D-Finite functions may be evaluated in quasi-linear time and linear space
- Proof based on Chudnovsky & Chudnovsky's binary splitting algorithm
  + effective error bounds
- Prototype implementation available (pure Maple)



Current & future work

- Make it practical!
- Seems to require a more careful / lower-level implementation

Credits

- Some icons used in this document are from the Oxygen icon set (`http://oxygen-icons.org`), licensed under the GNU Lesser General Public Licence v. 3.